PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : H04M 11/06, H04L 29/06

A1

(11) International Publication Number: WO 99/65219

(43) International Publication Date: 16 December 1999 (16.12.99)

(21) International Application Number: PCT/US99/13184

(22) International Filing Date: 10 June 1999 (10.06.99)

(30) Priority Data:
60/088,865 11 June 1998 (11.06.98) US
09/321,902 28 May 1999 (28.05.99) US

(71) Applicant: iREADY CORPORATION [US/US]; Suite 200, 2903 Bunker Hill Lane, Santa Clara, CA 95054 (US).

(72) Inventors: JOHNSON, Michael, Ward; 437 Ely Road North, Petaluma, CA 94954 (US). MINAMI, John, Shigeto; Suite 428, 330 Elan Village Lane, San Jose, CA 95134 (US). KOYAMA, Ryo; 343 Hawthorne, Palo Alto, CA 94301 (US).

(74) Agents: GLENN, Michael, A. et al.; Glenn Patent Group, 125 Lake Road, Portola Valley, CA 94028 (US).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

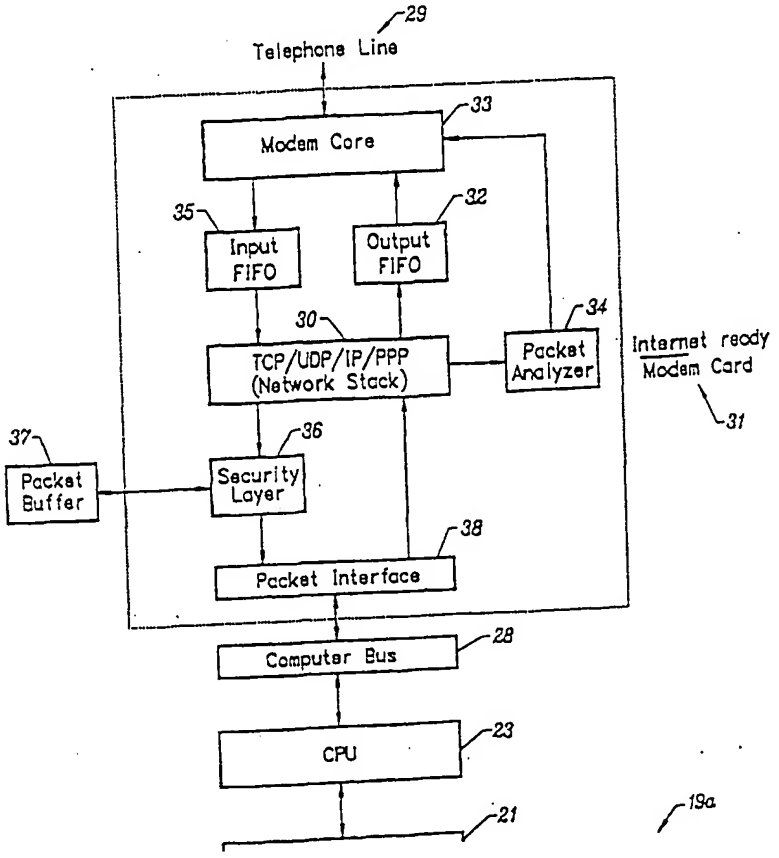
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: TCP/IP/PPP MODEM

(57) Abstract

An Internet network protocol stack, along with special logic, is embedded in hardware, thereby enabling a modem to become Internet-ready. As a result, the modem offloads much of the network protocol processing from the main CPU and improves the overall performance of the communication system.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania
AM	Armenia
AT	Austria
AU	Australia
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BJ	Benin
BR	Brazil
BY	Belarus
CA	Canada
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Côte d'Ivoire
CM	Cameroon
CN	China
CU	Cuba
CZ	Czech Republic
DE	Germany
DK	Denmark
EE	Estonia
ES	Spain
FI	Finland
FR	France
GA	Gabon
GB	United Kingdom
GE	Georgia
GH	Ghana
GN	Guinea
GR	Greece
HU	Hungary
IE	Ireland
IL	Israel
IS	Iceland
IT	Italy
JP	Japan
KE	Kenya
KG	Kyrgyzstan
KP	Democratic People's Republic of Korea
KR	Republic of Korea
KZ	Kazakstan
LC	Saint Lucia
LI	Liechtenstein
LK	Sri Lanka
LR	Liberia
LS	Lesotho
LT	Lithuania
LU	Luxembourg
LV	Latvia
MC	Monaco
MD	Republic of Moldova
MG	Madagascar
MK	The former Yugoslav Republic of Macedonia
ML	Mali
MN	Mongolia
MR	Mauritania
MW	Malawi
MX	Mexico
NE	Niger
NL	Netherlands
NO	Norway
NZ	New Zealand
PL	Poland
PT	Portugal
RO	Romania
RU	Russian Federation
SD	Sudan
SE	Sweden
SG	Singapore
SI	Slovenia
SK	Slovakia
SN	Senegal
SZ	Swaziland
TD	Chad
TG	Togo
TJ	Tajikistan
TM	Turkmenistan
TR	Turkey
TT	Trinidad and Tobago
UA	Ukraine
UG	Uganda
US	United States of America
UZ	Uzbekistan
VN	Viet Nam
YU	Yugoslavia
ZW	Zimbabwe

TCP/IP/PPP Modem

BACKGROUND OF THE INVENTION

TECHNICAL FIELD

The invention relates to combining a network stack with a modem core for use in both computer and non-computer applications. More particularly, the invention relates to an Internet-aware modem which combines any number of point-to-point devices with the network protocols necessary to communicate on the Internet, where these devices include various speed traditional modems from 2400 kbps to 56 kbps, ISDN modems, newer xDSL modems, and digital cellular modems.

DESCRIPTION OF THE PRIOR ART

Computer modems were developed in a time when most connections were made to proprietary online services, interactive terminals, bulletin board services (BBSs), or corporate network systems. As such, it was necessary to implement connection protocols in software because there existed at the time a number of such protocols. These protocols included x-modem, y-modem, z-modem, kermit, and interactive character based interfaces.

Today, with popularity of the Internet, a vast majority of modems are now used exclusively to connect with ISP's, which in turn connect the user to the Internet. Therefore, there is now a predominant set of connection protocols. Such protocols are used for most modem connections. Accordingly, there is a real need and advantage in designing a modem that is Internet-ready.

The connection protocols used by the Internet and their hierarchical relationship are shown in Fig. 1. These protocols include TCP 10, IP 11, UDP 13, and PPP 12. Optimizing a modem for use with the Internet offers many advantages including reduced transmission latency, reduced servicing

requirements, lower processing requirements from the system's CPU, and optimized transmission rates.

Current computer systems treat a modem subsystem as a serial device. A block diagram of an existing system is shown in Fig. 2. In such system, an Internet application, such as a Web browser 21, is run in software 19 on the main CPU 22. This application, in turn calls upon the computer network stack 23, which is also implemented in software. The network stack implements the TCP, IP, UDP, and PPP protocols. Once the data have been processed, the resulting packets are sent by the CPU via the computer bus 28 to a serial port interface 27 in the modem system 30. The modem system, for example a modem card 18, is seen as a serial I/O device by the host processor. These devices usually accept data in byte quantities and place them in an outgoing FIFO 24. These FIFOs can be anywhere from 8 bytes to 64 bytes. The CPU normally writes a fixed number of bytes, then waits for the serial I/O device to notify it that all the data have been sent and that it is ready to accept more data. This notification is usually done via system interrupts. The packet data, after it is written into the FIFO, is fed to the modem core 25 at the outgoing data rate and thence to the telephone line 29.

For received data, the modem first places all incoming packets into the input FIFO 26. The device can then be configured to interrupt the host CPU when any data are available or when the received data reaches some level (*i.e.* 16 bytes). When notified, the CPU reads all the data in the input FIFO, and stores the data temporarily in a buffer in system memory (not shown). The bottom protocol, PPP (see Fig. 1) can start to process the data, but it cannot pass up the data to the next layer until the entire packet is received.

Once the whole packet is received, the PPP portion of the software network stack passes the data up to the second protocol (IP). The IP software then processes the IP header and, after verifying the header checksum, passes the packet to the TCP handler. The TCP handler then checks its checksum, and

passes the data on to the appropriate application, as specified by the PORT number in the TCP header.

Because most modems in computers today are used to connect to the Internet, it makes it economically feasible and practical to optimize a modem for this environment. What this entails is embedding in the modem system, the ability to handle the necessary network protocols and use the knowledge of the protocols to tune the transmission characteristics of the modem. This is the same rationale behind the popularity of Window's accelerator graphics cards. Because graphic chip manufacturers know that a vast majority of PC's today run the Microsoft Windows ® operating system, they fine-tune their architectures to enhance the performance in this environment. This would not be practical if there were a number of operating systems with different graphic APIs, each with a significant portion of the market place. However, with the one over-riding standard, most graphic card manufactures have chosen to optimize their hardware for the Windows environment, even though today's Pentium class processors are very capable of handling the graphic chores without external support. This is because the function is required in most circumstances, and it is advantageous to offload the host processor so that it has that much more MIPs to run standard applications.

A similar situation now exists in the modem card market. It would therefore be advantageous to embed the Internet network protocol stack, along with special logic, thereby enabling the modem device to become Internet-ready, such that the modem system offloads much of the network protocol processing from the main CPU, while improving the overall performance of the communication system.

SUMMARY OF THE INVENTION

The invention embeds the Internet network protocol stack, along with special logic, thereby enabling the modem device to become Internet-ready. As a result, the modem system offloads much of the network protocol processing

from the main CPU and improves the overall performance of the communication system. The invention provides an Internet-aware modem which combines any number of point-to-point devices with the network protocols necessary to communicate on the Internet, where these devices include various speed traditional modems from 2400 kbps to 56 kbps, ISDN modems, newer xDSL modems, and digital cellular modems.

Sending Data

In a system equipped with an Internet-ready modem, the Internet application first sets up the socket parameters. These include the destination port number, the type of connection (TCP/UDP), the TOS (Type-Of-Service) requirements, and the destination IP address. When the network stack on an Internet-ready modem card gets this information, it attempts to start a connection by sending out a SYN packet. This packet is passed to the IP engine, which attaches the IP header and calculates the IP header checksum. The packet is then passed to the PPP handler which attaches the PPP header, escapes the data, and appends the PPP checksum. After PPP encapsulation, the resulting network packet is sent through the output FIFO to the modem core. For this packet, the TCP engine indicates to the packet analyzer block that it is a SYN packet. The packet analyzer then indicates to the modem that this is a stand alone packet and that it can be sent immediately. Upon receiving this information, the modem sends the network packet out without first waiting the normal 50 ms to see if additional information is forthcoming.

After the destination socket sends a return SYN-ACK packet, an ACK packet is sent from the modem card. This packet follows the same steps as those for the SYN packet. At this point, the socket connection is up, and the application software (such as a Web browser) is notified. The application can now send its data directly to the modem in a data packet format. In this example, where the application is a Web browser, the application can send an HTTP request

directly to the modem system via a packet interface as opposed to the serial port I/O interface in a regular modem system. DMA style data transports can be used for this purpose. In this method, a data byte count is programmed into the packet interface. Data can then be automatically transferred from memory into a modem card without further intervention from the host CPU. After all the data have been transferred, an interrupt from the modem card can be sent to the host CPU indicating that the data transfer is complete.

As the data arrive at the modem card, they are sent (in this example) to a TCP data output buffer. After all the data are received or when the maximum data size per packet has been received, the TCP block begins calculating the checksum. The packet is encapsulated in the same way as the SYN packet. In parallel to this, the TCP engine indicates to the packet analyzer block that the destination port for the packet is 80, which is the well-known HTTP port. The packet analyzer then knows that there are no more data and, again, the modem should send the current packet immediately.

Receiving Data

When receiving network packets, the data are sent from the modem core through the input FIFO to the PPP engine. which parses the PPP header, unescapes the data, and starts a running check on the packet for checksum calculations. If the engine determines that the encapsulated data is an IP packet, it enables the IP engine, and all data past the PPP header is forwarded to the IP engine. The IP engine parses the IP header, checks the checksum, and if it determines that the encapsulated protocol is TCP, then it sends all data past the IP header to the TCP engine. The TCP engine then parses the TCP header and sends the data portion to the security layer. If the data is HTML data, it can be passed through a ratings check that parses out rating tags of pages. If the page has a rating below or equal to the modem cards setting, then the data are allowed to pass. If the rating exceeds the setting on the card, a message indicating so is passed on instead. If the page

contains no ratings, a bit can be set to either pass or block the page. All non-HTML data are passed directly to the TCP data buffer.

As the data are being written into the buffer, a running count is kept to see how much data have been received. At the end of the network packet, if the PPP checksum indicates that the entire packet was received without errors, then an interrupt can be generated to the host CPU. The application can then read the received data count, and program a DMA transfer to transfer data from the TCP buffer into main memory.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram showing connection protocols used by the Internet and their hierarchical relationship;

Fig. 2 is a block diagram showing a typical modem subsystem;

Fig. 3 is a block schematic diagram showing an Internet-ready modem system according to the invention;

Fig. 4 is a block schematic diagram showing a modem having a full network stack according to t he invention;

Fig. 5 is a block schematic showing a PPP function according to the invention;

Fig. 6a is a block schematic diagram showing a prior art modem and network card;

Fig. 6b is a block schematic diagram showing a modem according to the invention and an enhanced Ethernet network card;

Fig. 7 is a schematic diagram showing fields used in a PPP protocol packet to determine latency according to the invention;

Fig. 8 is a block schematic diagram showing optimization based on TOS according to the invention;

Fig. 9 is block schematic diagram showing an optimization based on destination port according to the invention;

Fig. 10 is a block schematic diagram showing an optimization based on packet state according to the invention;

Fig. 11 is a block schematic diagram showing combined latency tables according to the invention;

Fig. 12 is a block schematic diagram showing an enhanced modem system according to the invention;

Fig. 13 is a block schematic diagram showing received packet data flow through an HTML sniffer according to the invention; and

Fig. 14 is a block schematic diagram showing an HTML sniffer according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

The invention provides an Internet-aware modem which combines any number of point-to-point devices with the network protocols necessary to communicate on the Internet, where these devices include various speed traditional modems from 2400 kbps to 56 kbps, ISDN modems, newer xDSL modems, and digital cellular modems. The invention embeds the Internet network protocol stack, along with special logic, thereby enabling the modem device to become Internet-ready. As a result, the modem system offloads much of the network protocol processing from the main CPU and improves the overall performance of the communication subsystem.

The invention provides several technologies that enhance modem performance and efficiency when used with Internet protocols. These technologies include:

- Block Based Communication,
- Latency Optimization,
- Reduced Processing Power,
- Reduced Energy Requirements, and
- Security Enhancements.

Data Transfer Overhead Reduction

Modems today all communicate via serial ports. This serial communication has several performance disadvantages over other communications devices, such as Ethernet adapters, which communicate to the host CPU by sending and receiving blocks of data via DMA (Direct Memory Access).

Serial Communications

Serial communication using older serial hardware (*e.g.* UARTs) require an interrupt for every character sent or received. This causes so much overhead that it is not possible to communicate at speeds over 19200bps without dropping data on most computer systems. Second generation serial hardware (*e.g.* the 16550 UART and its derivatives) are able to buffer up to 16 bytes on send and receive and can delay interrupts until a buffer reaches a certain level. This reduces the number of interrupts required to transfer data to and from the modem. This allows modern modems to achieve serial data rates of 56000bps to 230400bps without losing data on most CPU's.

Although high speed serial communication using second generation serial hardware is now possible, at these higher data rates serial communications

can cause a noticeable degradation in system performance, especially when the computer system is running Internet enabled action games or multi-media communication programs. Thousands of interrupts each second are required at the higher data rates (see Table 1 below.)

Table 1. Serial Interrupt Requirement Example

Given a 16550 UART with interrupts set at: 16 bytes transmit trigger 14 bytes receive trigger Receive interrupts: 230400bps/9bits per byte=25600Bps/14 bytes per interrupt = 1828 interrupts per second. Transmit Interrupts: 230400bps/9bits per byte = 25600Bps/16 bytes per interrupt = 1600 interrupts per second.

Each interrupt requires an interrupt service routine to read and write data to/from the serial hardware and to/from the Computers Operating System. These interrupt routines read and write to I/O ports that are not efficient use of system resources.

Optimizing Using Block Transfers

There has been talk about a next generation UART that would have either 32 or 64 byte buffers to reduce the number of interrupts and load on a system. While this may be helpful, the inventor's have recognized that the correct solution is to optimize the modem for Internet protocols and transfer data as blocks of data packets.

Block based Communication to/from modems has not been previously considered because the Internet protocols designed to interface a modem/serial device are serial based (SLIP/PPP) and are traditionally implemented in the computer system's operating system. Using the herein described invention, it is advantageous to move the network stack protocols from the computer operating system into the modem. The layers of the network protocol that can be implemented in a hardware device in accordance with the invention can range from just the PPP layer all the way to the IP, TCP, and UDP layers.

Example: Block Based Interrupt Requirement

Assuming an average 100-byte IP packet:
25600 Bps / 100 = 256 Interrupts per second.

As is shown in the above example, transferring blocks instead of chunks of characters, as in the serial solution, reduces the number of interrupts required. An added benefit is that the transfer routine can use direct memory access (DMA) to pass data instead of slower I/O port pumping. Currently serial port hardware and software do not support DMA.

This enables the development of software that can produce faster, more efficient data handling routines because the host CPU is relieved of handling transferring of individual bytes of data.

Reduced Latency

Once the modem knows that it is dealing with IP packets, it has knowledge of where data units start and stop. Modem protocols, such as V.42 and V.42bis, can be optimized to take advantage of this knowledge.

V.42 has an outdated view of what kind of data are being carried over a modem, but if the network stack is embedded alongside a modem core, it can

indicate to the modem core the end of data units. The modem core can then segment its V.42 packets accordingly. This reduces latency on retransmission because retransmission would have a higher probability of affecting only one IP packet. Knowledge of the end of a packet can also be used to reduce the waiting for more data timeout that the V.42 protocol introduces. If it is known that the last byte received is an end of a physical block of data, the modem need not wait around, or can reduce the time it waits, for more data to be sent to it before it sends that data it has ready to send.

This aspect of the invention can be extended to perform IP and TCP packet snooping to optimize further what gets sent when and how long the V.42 protocol waits before compressing and sending the current block. For example, the TOS field in the IP header can be used to determine the amount of latency used in the transmission of the packet. If the packet is a high priority packet the system may decide to send the packet immediately, not checking to determine if there are more packets ready to send. The system can wait varying lengths of time for more data based on this TOS information.

Additional latency optimization can be achieved by checking the TCP header. If the SYN flag is set in the TCP header, then the data should also be sent immediately because nothing more can happen until the SYN-ACK is returned from the other side of the connection.

Also, unlike previous advances to modem technology that required that the same technical advances be on both the receiving and sending ends of the communications link, the herein disclosed modem can operate and is compatible with all existing modems today. Therefore, it is possible to gain performance increases by updating only one end of the connection. In this way, adoption of the technology is not dependent on changes in ISP infrastructure.

Reduced Processing Power Required from Host CPU

By embedding the network stack along with the modem core, a marked reduction in processing power that is required to connect and communicate on the internet is possible. This allows small, low power, low cost devices to communicate via the Internet using modems. Examples of these types of devices include game consoles, PDAs, toys, and other consumer electronic and mobile electronic appliances.

Processor Power Reduction of PPP

The PPP protocol requires that packets include a CRC appended at the end of each packet. This calculation requires what could be a significant amount of processing power on low-end processors. Other aspects of the protocol, such as escaping data and parameter negotiations, require memory accesses for each byte when implemented as a software solution. By implementing PPP in the modem, all negotiations can be kept local to the modem subsystem, thereby relieving system bus traffic and processing overhead.

Processor Power Reduction of Embedding IP

Embedding IP offloads the header checksum calculations from the host processor. It also keeps ICMP echo packet processing local to the modem card. This protocol is used for PING applications.

Processor Power Reduction of Embedding UDP

Though it is possible to use UDP as an encapsulation device requiring little processing power, when used with checksums, UDP can require some

processor resources. It should be noted here that most thin Internet clients are based on IP/UDP to transmit their data.

Processor Power Reduction of Embedding TCP

TCP is a much more complicated protocol than UDP and, thus, requires much more processing power. The TCP has many states and requires checksums to be preformed on packets. For embedded products that require TCP support the invention described herein provides a way to offload all the complexity and processor requirements onto a dedicated hardware circuit.

Portability of Solution

By embedding the network stack inside of the modem subsystem, the same modem system can now be used across multiple computing and system platforms. Because no porting of any network stack software is required, moving the modem into different systems becomes very easy. This is especially important in the embedded systems market that does not have one or two main OS's but instead is made up of a number of different OS's, RTOS's, and in some cases, no OS. The embedded systems market is also characterized by being made up of a number of incompatible processors. This lack of an overriding standard also favors a highly portable network solution.

Reduced Energy Requirements

The herein described invention is also very efficient in terms of energy requirements. A highly optimized state machine reduces by two orders of

magnitude the clock rate required to perform the functionality of the Internet suite of protocols. This translates into extended battery life and less heat generated by products designed with the herein described invention.

Security Enhancements

By implementing the network stack in hardware along with the modem, the invention provides a very secure, unhackable network stack. This is due to the hardware architecture implementation that disregards any packet received unless there is already a socket connection set up for it. Furthermore, the packets do not get past the modem card, therefore making any interaction between unrequested packets and software impossible.

In addition, by including an HTML packet sniffer, it is possible to decode HTML rating tags. The packet sniffer interprets bytes in the packet buffer, and can be set up to pass only those pages that are within its preset rating level. For pages that exceed the programmed rating level, the HTML sniffer passes up a failed retrieval message, and does not allow the HTML content past the modem subsystem. This turns the modem into a content driven mini-firewall.

For those pages that do not include a rating tag, the sniffer can be configured either not to pass these pages at all, or to allow the pages to be passed. The rating level can be programmed only via board settings, *i.e.* hard wired. The advantage of implementing this solution in hardware is that for parents or anyone who wishes to filter out certain Web sites, it is impossible to get around this system without taking the modem card out of the system. With any software solution, the user could simply load a non-filtering browser or disable a certain plug-in, and the filter would be bypassed. By providing the filter in hardware on the modem card, there is no way to bypass the function.

System Implementation

Fig. 3 is a block schematic diagram of an Internet-ready modem system.

Sending Data

In a system equipped with an Internet-ready modem 31, the Internet application first sets up the socket parameters. These include the destination port number, the type of connection (TCP/UDP), the TOS (Type-Of-Service) requirements, and the destination IP address. When the network stack 30 on the Internet-ready modem 31 gets this information, it attempts to start a connection by sending out a SYN packet. This packet is passed to the IP engine within the network stack, which attaches the IP header and calculates the IP header checksum. The packet is then passed to the PPP handler within the network stack which attaches the PPP header, escapes the data, and appends the PPP checksum. After PPP encapsulation, the resulting network packet is sent through the output FIFO 32 to the modem core 33. For this packet, the TCP engine indicates to the packet analyzer block 34 that it is a SYN packet. The packet analyzer 34 then indicates to the modem that this is a stand alone packet and that it can be sent immediately. The modem upon receiving this information sends the network packet out without waiting the normal 50 ms to see if additional information is forthcoming.

After the destination socket sends a return SYN-ACK packet, an ACK packet is sent from the modem card. This packet follows the same steps as those for the SYN packet. At this point, the socket connection is up, and the application software (such as a Web browser 21) is notified. The application can now send its data directly to the modem in a data packet format.

In the example shown in Fig. 3, where the application is a Web browser, the application can send an HTTP request directly to the modem system via a packet interface 38 as opposed to the serial port I/O interface in a regular

modem system (see Fig. 2). DMA style data transports can be used for this purpose. In this method, a data byte count is programmed into the packet interface. Data can then be automatically transferred from memory into a modem card without further intervention from the host CPU. After all the data have been transferred, an interrupt from the modem card can be sent to the host CPU 23 indicating that the data transfer is complete.

As the data arrive at the modem card, they are sent (in this example) to a TCP data output buffer. After all the data are received or when the maximum data size per packet has been received, the TCP block begins calculating the checksum. The packet is encapsulated in the same way as the SYN packet. In parallel to this, the TCP engine in the network stack indicates to the packet analyzer block that the destination port for the packet is 80, which is the well-known HTTP port. The packet analyzer then knows that there are no more data and, again, the modem should send the current packet immediately.

Receiving Data

When receiving network packets, the data are sent from the modem core 33 through the input FIFO 35 to the PPP engine in the network stack, which parses the PPP header, unescapes the data, and starts a running check on the packet for checksum calculations. If the engine determines that the encapsulated data is an IP packet, it enables the IP engine in the network stack, and all data past the PPP header is forwarded to the IP engine. The IP engine parses the IP header, checks the checksum, and if it determines that the encapsulated protocol is TCP, then it sends all data past the IP header to the TCP engine in the network stack. The TCP engine then parses the TCP header and sends the data portion to the security layer 36. If the data is HTML data, it can be passed through a ratings check that parses out rating tags of pages. If the page has a rating below or equal to the modem cards setting, then the data are allowed to pass. If the rating exceeds the setting on the card, a message indicating so is passed on instead. If the page contains

no ratings, a bit can be set to either pass or block the page. All non-HTML data are passed directly to the TCP data buffer 37.

As the data are being written into the buffer, a running count is kept to see how much data have been received. At the end of the network packet, if the PPP checksum indicates that the entire packet was received without errors, then an interrupt can be generated to the host CPU 23. The application can then read the received data count, and program a DMA transfer to transfer data from the TCP buffer into main memory.

Features of the Invention

The following discussion describes various features of the invention:

1. Modem as a Block Device.

2. Latency Optimization based on packet parameters:

   a) Basic End of Packet Optimization,

   b) Optimization based on IP TOS flag,

   c) Optimization based on UDP/TCP port numbers,

   d) Optimization based on TCP State, and

   e) Latency Table.

3. Modem as a complete Internet Access Device:

a) Partial Stack solutions:

i) PPP/IP,

ii) PPP/IP/ICMP, and

iii) PPP/IP/ICMP/UDP; and

b) Complete Stack Solutions (PPP/IP/ICMP/UDP/TCP).

4. Enhanced Security and HTML filtering in an Internet enabled Modem.

The following discussion describes the above features in more detail.

Modem as a Block Device

Depending on the network layers included with the modem hardware, different data formats are sent to the modem subsystem. In any of the implementations, however, DMA transfers can be used to optimize CPU overhead required for the transfers.

Refer to Fig. 4 for the following discussion. In the implementation where the entire network stack 40 is included with the modem card 41, only the application 42 data need be transferred. Software applications 44 communicate with the modem card via a socket API 43.

At a minimum, just PPP can be added to the modem. The function of PPP is to transform IP packets (blocks of data) into a serial stream so that it can be transported over a serial device (see Fig. 5). The PPP protocol is also responsible for negotiating the parameters that are used to transmit data over the serial link (*e.g.* compression schemes and escaped bytes). With the PPP

function 50 performed inside the modem, the communication between the modem and the IP protocol stack software operates in a manner that is similar to that in which an IP stack communicates with an Ethernet card.

As with an Ethernet card, packets are shipped from the IP protocol stack to the device driver. In the modem's case, the device driver is a simple interfacing software program that transfers blocks of data to and from the modem and the host computer. Compare the architecture of the prior art modem of Fig. 6a with that of a modem according to the invention, as shown in Fig. 6b.

This embodiment of the invention enables all of the efficiency benefits described above, and is an attractive solution because it can be implemented by adding only a minimum amount of extra logic to existing modem chip sets, and because it requires as little as 512 bytes of memory for support. This makes it very cost affordable to add to any existing modem.

Latency Optimization

Traditional modems have no knowledge of the type of data they carry and their protocols are optimized for interactive character based interfaces. The traditional modem protocols have a built in 50ms delay before sending information. This delay is present because the modem has no idea of where the data stops and starts, so it waits until it knows no more data are going to be sent.

With the popularity of the Internet, almost all modems today are used to connect to the Internet. Using this knowledge and the information on when the packets begin and end can help optimize modem transmissions. This optimization can reduce the amount of time it takes to move an Internet protocol packet from one modem to another by reducing or eliminating the 50ms delay built into the modem protocols. This feature of the invention can be extended using different parts of the network packet to make decisions

whether there should be any delay or how long it should be before the modem processes a packet. The hardware module that handles this optimization is the packet analyzer block 34 (see Fig. 3).

Basic End of Packet Optimization

At the most basic level, one could use the knowledge of the end of the PPP packet to tell the modem protocols to wait a small amount of time before sending the packet (see Fig. 7). In the traditional modem model, if there were not enough bytes to make a modem protocol frame (such as V42) the modem would wait for more data, up to 50ms, before timing out and sending the packet. With knowledge of the end of the PPP packet and the encapsulated protocol, the modem could expedite the sending of the packet knowing that it has a complete packet.

This algorithm is useful for PPP sub-protocols, such as LCP (Link Control Protocol), PAP (password Authentication Protocol), CHAP (Challenge Handshake Authentication Protocol); and NCP (Network Control Protocol). With these and similar PPP sub-protocols, packets transmitted are stand-alone in that all information is contained within a single packet. Also, in most cases after the packet is sent, no further data are sent because the device is waiting for a response from the other device. Therefore, if the packet analyzer detects that a PPP packet contains a PPP sub-protocol, when it detects the PPP FCS field it can instruct the modem to wait only 2 ms before sending the data, instead of the normal 50 ms. The reason to wait a minimum of at least 2 ms is that in the transition between the LCP, Authentication, and NCP phases of the PPP negotiations, back to back packets can be sent out. However, there would never be more than two back to back packets, and the second packet always follows immediately within 2 ms of the first packet.

Further optimization can occur by looking at the command code of the PPP sub-protocol packet. An example matrix of command types and the corresponding latency setting are shown in the Table 1 below.

Table 2. Matrix of Command Types and Corresponding Latency Setting

Protocol	Command Code		Latency Setting
LCP	0x01	Configuration Request	0 ms
	0x02	Configuration Ack	2 ms
	0x03	Configuration Nak	0 ms
	0x04	Configuration Reject	0 ms
	0x05	Termination Request	0 ms
	0x06	Termination Acl	0 ms
	0x07	Code Reject	2 ms
	0x08	Protocol Reject	2 ms
	0x09	Echo-Request	2 ms
	0x0A	Echo-Reply	2 ms
	0x0B	Discard-Request	2 ms
PAP	0x01	Authentication-Request	0 ms
CHAP	0x02	Challenge Response	2 ms
NCP	0x01	Configuration Request	0 ms
	0x02	Configuration Ack	2 ms
	0x03	Configuration Nak	0 ms
	0x04	Configuration Reject	0 ms
	0x05	Termination Request	0 ms
	0x06	Termination Acl	0 ms

Optimizing Based On IP Header Fields

Optimizing Based on the TOS Field

FOR IP, TCP, and UDP packets, a more intelligent decision on how long to wait or when to send packets can be determined by examining the type of service (TOS) field in the IP header. The TOS field describes the priority and reliability requested for the packet. The properties that are settable for the TOS field are Minimize Delay, Maximize Throughput, Maximize Reliability, and Minimize Cost. More than one of the TOS flag properties can be set at

one time. This information can be used to set variable waiting time delays or to send what is in the modem buffer immediately.

Fig. 8 shows how it is possible to look inside the TOS flag 80 for a minimize delay property and use that information and the information on when the IP packet ends to tell the modem protocols to send the packet immediately.

Optimizing Based on the Encapsulated Protocol

Another optimization that can be performed based on IP header fields is to base the latency setting on the protocol field. Most ICMP and IGMP packets are self contained therefore minimum wait times are needed after they are sent. After the packet analyzer determines that the IP packet contains either an IGMP or ICMP packet, it signals to the modem core to send the packet immediately.

Optimization Based on Packet Ports

Certain kinds of Internet services have information distributions that require just one packet of data to be sent and received. With these types of services it is optimal always to send a packet immediately without waiting for more data. Other Internet services have yet other packet distribution patterns that could be optimized for UDP and TCP, the major protocols that are used on top of IP. Both use ports to describe services. Fig. 9 provides an example of how the port information carried in both UDP and TCP is used to optimize modem latency.

The latency table 90 contains a table of ports and the amount of time to wait after the end of the packet for more data. An example of optimizing using this method is the DNS application. In this application, the entire data portion of the message can easily fit in one Internet packet. Therefore, if after

examining the destination port in the UDP header it is determined that it is a DNS packet, the packet can be sent immediately because there are not any further packets coming. Table 3 below provides an example protocol-port latency table.

Table 3. Latency Table - Example I

Protocol	Port	Application	Wait Time
TCP	0x07	Echo	0 ms
	0x17	Telnet	50 ms
	0x19	SMTP	10 ms
	0x50	HTTP	0 ms
	0x6E	POP3	0 ms
UDP	0x07	Echo	0 ms
	0x53	DNS	0 ms

Latency Optimization Based on Packet State

TCP is a state-based protocol and certain states have well known properties that can be used for latency optimization. An example is the three-way handshake with which all TCP connections start. The first few packets of this transaction are small packets that must be sent before any further communications can take place. The invention can noticeably improve the time this connection process takes. This can be a very noticeable improvement, especially when operating software that connects many TCP connections at one time during one transaction, as with a Web browser.

In Fig. 10, information from the IP header (TCP protocol type) 100 and the TCP state 101 are used to look up a latency value to pass through to the modem protocol at the end of the packet. Table 4 below provides an example latency table using this information.

Table 4. Latency Table - Example II

TCP Flags						Wait Time
URG	ACK	PSH	RST	SYN	FIN	
X	1	0	0	X	0	2 ms
X	X	X	X	1	X	2 ms
X	X	X	X	X	X	2 ms
X	X	1	X	X	X	50 ms

X = Don't Care

The Latency Table

Latency tables are state machines that have a number of inputs that are triggered by a packet's characteristics. From these inputs the latency table produces an optimized latency value for the modem protocol, effectively optimizing each packet as it passes through the system. A block schematic diagram of the latency tables is shown in the Fig. 11, in which the information discussed above is combined.

The IP latency resolver 110 takes the inputs from the IP sub-protocol latency table 111 and the IP TOS field latency table 112 and selects the lower of the two values. The TCP latency resolver 113 performs a similar function for the destination port latency table 114 and the TCP state latency table 115. The IP latency resolver and TCP latency resolver outputs are muxed 117 to produce a combined latency value therefor. (The mux selection is determined by the protocol field in the IP header as parsed by the IP Sub-Protocol Latency Table III.) A value is also provided by the PPP latency table 116. This value is muxed with the muxed 117 value of the IP latency resolver and TCP latency resolver. The mux selection for mux 118 is determined by the protocol field in the PPP header as parsed by the PPP latency table 116. The final latency value is then sent to the modem subsystem.

Modem as a Complete Internet Access Device

The extended Internet modem model is a stand alone, embeddable, highly integrated communications product that can Internet enable almost any electronic device (see Fig. 12). This solution has a number of advantages over the processor based solutions discussed above. More specifically, this architecture allows the addition of Internet support to non-computer applications, such as game consoles and VCR's. It is also very useful for those devices that have limited memory footprints and do not need network support all the time. An example of this is Palm Pilot type devices, where the only time the added network support is needed is when the modem is used. An advantage of the invention is that it is not necessary to waste memory resources on features that are not used all the time.

Enhanced Security Benefits

As stated above, one security benefit of having a hardware based network stack is that only those packets received that are destined for a preconfigured socket connection are allowed past the modem subsystem. All other packets are filtered out at the hardware level, making any interaction between these packets and software impossible. Also, with the addition of the HTML sniffer, V-Chip like filtering can be provided that cannot be easily circumvented. Block diagrams of the HTML sniffer are shown in Figs. 13 and 14.

In Fig. 13, a packet 138 is received at the TCP engine 139. The packet is destined for a specific socket. The HTML packet sniffer 144 has a preset rating 146 that is applied to the packet to determine if the packet is to be placed in the received packet buffer 145.

As shown in Fig. 14, within the HTML packet sniffer, the HTTP response parser 140 takes received packets from the socket 141, and interprets the HTTP header to determine if the data content contains valid HTML data. If so, it enables the HTML rating decoder 142, which begins to parse the HTML data for rating tags. The HTML decoder writes all received data to the received packet buffer 145 (including the HTTP header), and at the same time parses tags. If it detects a rating tag, it compares the page's rating to the card's preset rating level. If it passes, then the page continues to be stored in the receive buffer. If the page fails, then all further data are suppressed, the memory buffer is reset to the point prior to receiving the current packet, and a reject message is stored in memory. If the page contains no ratings at the head of the page, the card can either be configured to pass the page or reject the page.

Although the invention is described herein with reference to the preferred embodiment, one skilled in the art will readily appreciate that other applications may be substituted for those set forth herein without departing from the spirit and scope of the present invention. Accordingly, the invention should only be limited by the Claims included below.

CLAIMS

1. A modem, comprising:

a modem core; and

a network stack embedded in hardware which executes network protocols to allow said modem to communicate on an electronic network.

2. The modem of Claim 1, wherein said network stack comprises:

an Internet network protocol stack.

3. The modem of Claim 1, wherein said network stack sends and receives blocks of data.

4. The modem of Claim 1, wherein network protocols that are implemented in said network stack include any of the protocols required by PPP, IP, TCP, and UDP communications layers.

5. The modem of Claim 1, wherein said network stack supports direct memory access (DMA) operations to optimize CPU overhead required for data transfers.

6. The modem of Claim 1, wherein said network stack performs IP and TCP packet snooping.

7. The modem of Claim 1, wherein the TOS field in an IP header is used by said network stack to determine the amount of latency used in the transmission of a packet.

8. The modem of Claim 1, wherein said network stack performs latency optimization by checking a TCP header.

9. The modem of claim 1, wherein all protocol negotiations are kept local to said modem by said network stack.

10. The modem of Claim 1, wherein said network stack disregards any packet received unless there is already a socket connection set up for it.

11. The modem of Claim 1, further comprising:

a packet sniffer for decoding HTML rating tags, wherein said packet sniffer only passes those pages that are within a preset rating level.

12. The modem of Claim 11, wherein said packet sniffer is configured either not to pass unrated pages that do not include a rating tag at all, or to allow said unrated pages to be passed.

13. The modem of Claim 11, wherein a rating level can be programmed only via hard wired settings.

14. The modem of Claim 1, wherein software applications communicate with said modem via a socket API.

15. A modem, comprising:

a modem core; and

a network stack embedded in hardware which executes network protocols to allow said modem communicate on an electronic network, wherein said network stack comprises an Internet network protocol stack.

16. The modem of Claim 15, wherein said network stack further comprises:

a PPP stack for transforming IP packets into a serial stream for transport over a serial link, and for negotiating parameters that are used to transmit data over said serial link

17. The modem of Claim 15, further comprising:

a packet analyzer module for using different parts of a network packet to make decisions whether there should be a delay and, if so, how long it should be before said modem processes a packet.

18. The modem of Claim 15, further comprising:

a packet analyzer module for using knowledge of the end of a PPP packet to tell said modem to wait a minimum amount of time before sending a packet.

19. The modem of Claim 18, wherein said packet analyzer module detects PPP sub-protocols, including any of LCP (Link Control Protocol), PAP (password Authentication Protocol), CHAP (Challenge Handshake Authentication Protocol), and NCP (Network Control Protocol), and uses this information to determine the amount of latency used in the transmission of a packet.

20. The modem of Claim 19, wherein said packet analyzer module examines a command code of a PPP sub-protocol packet, and uses this information to determine the amount of latency used in the transmission of a packet.

21. The modem of Claim 18, wherein said packet analyzer module examines a type of service (TOS) field in an IP header to determine the amount of latency used in the transmission of packets.

22. The modem of Claim 21, wherein said TOS field describes priority and reliability requested for a packet.

23. The modem of Claim 21, wherein properties that are settable for said TOS field include any of Minimize Delay, Maximize Throughput, Maximize Reliability, and Minimize Cost.

24. The modem of Claim 23, wherein more than one of said TOS flag properties can be set at one time.

25. The modem of Claim 18, wherein said packet analyzer module examines a TOS flag for a minimize delay property, and wherein said network stack uses said minimize delay property and information on when an IP packet ends to instruct said modem to send a packet immediately.

26. The modem of Claim 18, wherein an optimization is performed based on IP header fields.

27. The modem of Claim 26, wherein a latency setting is based on a protocol field.

28. The modem of Claim 26, wherein port information carried in both UDP and TCP is used to optimize modem latency.

29. The modem of Claim 15, further comprising:

a latency table.

30. The modem of Claim 29, wherein said latency table comprises:

a table of protocols;

a table of ports; and

a table of amount of time to wait after the end of a packet for more data.

31. The modem of Claim 29, wherein information from any of an IP header and a TCP state is used to look up a latency value to pass through to said modem protocol at the end of said packet.

32. The modem of Claim 29, wherein said latency table is a state machine that has a plurality of inputs that are triggered by a packet's characteristics, wherein said latency table produces an optimized latency value for a modem protocol from said inputs, and wherein each packet is optimized as it passes through said modem.

33. The modem of Claim 29, further comprising:

an IP latency resolver that takes input values from an IP sub-protocol latency table and an IP TOS field latency table, and that selects a lower of said two values.

34. The modem of claim 29, further comprising:

a TCP latency resolver that takes input values from a destination port latency table and a TCP state latency table, and that selects a lower of said two values.

35. The modem of Claim 29, further comprising:

an IP latency resolver that takes input values from an IP sub-protocol latency table and an IP TOS field latency table, and that selects a lower of said two values; and

a TCP latency resolver that takes input values from a destination port latency table and a TCP state latency table, and that selects a lower of said two values;

wherein said IP latency resolver and said TCP latency resolver outputs are muxed to produce a combined latency value, with said mux being controlled by the protocol field within the IP header.

36. The modem of Claim 29, wherein a latency setting value is provided by a PPP latency table, with said value being determined by the PPP protocol field and the PPP command code field.

37. The modem of Claim 36, wherein said PPP latency table value is muxed with the muxed value of said IP latency resolver and said TCP latency resolver.

38. The modem of Claim 15, wherein said modem comprises:

a stand alone, embeddable, highly integrated communications product that can Internet enable a wide variety of electronic devices.

39. The modem of Claim 15, further comprising:

a TCP socket filter, wherein only those packets received at said modem that are destined for a preconfigured socket connection are allowed past said modem; and wherein all other packets are filtered out at by said socket filter at the hardware level.

40. The modem of Claim 15, further comprising:

an HTML packet sniffer which parses received HTML data.

41. The modem of Claim 40, wherein said HTML packet sniffer further comprises:

a preset rating that is applied to a packet to determine if said packet is to be placed in a received packet buffer.

42. The modem of Claim 40, wherein said HTML packet sniffer further comprises:

an HTTP response parser that takes received packets from a socket, and that interprets an HTTP header to determine if data therein contains valid HTML data and , if so, that enables an HTML rating decoder, which begins to parse said HTML data for rating tags.

43. The modem of Claim 40, wherein said HTML packet sniffer further comprises:

an HTML decoder that writes all received data to a received packet buffer, and at the same time parses tags;

wherein, if said HTML decoder detects a rating tag, it compares a corresponding page's rating to a preset rating level;

wherein, if said page passes, then said page continues to be stored in a receive buffer;

wherein if said page fails, then all further data are suppressed and a reject message is stored in memory; and

wherein if said page contains no ratings at a head of said page, said modem can be configured to either pass the page or reject said page based upon a user configuration.

44. A modem, comprising:

a modem core; and

a network stack embedded in hardware which executes network protocols to allow said modem to communicate on an electronic network, wherein said network stack comprises an Internet network protocol stack, wherein said network stack sends and receives blocks of data, wherein network protocols that are implemented in said network stack include any of the protocols required by PPP, IP, TCP, and UDP communications layers, wherein said network stack supports direct memory access (DMA) operations to optimize CPU overhead required for data transfers.

45. A method for producing a modem, comprising the steps of:

providing a modem core; and

embedding a network stack in hardware, said network stack executing network protocols to allow said modem to communicate on an electronic network.

46. The method of Claim 45, wherein said network stack comprises an Internet network protocol stack, wherein said network stack sends and receives blocks of data, wherein network protocols that are implemented in said network stack include any of the protocols required by PPP, IP, TCP, and UDP communications layers, wherein said network stack supports direct memory access (DMA) operations to optimize CPU overhead required for data transfers.

47. The method of Claim 45, further comprising the step of:
performing IP and TCP packet snooping.

48. The method of Claim 45, further comprising the step of:
using a TOS field in an IP header to determine the amount of latency used in the transmission of a packet.

49. The method of Claim 45, further comprising the step of:
performing latency optimization by checking a TCP header.

50. The method of claim 45, wherein all protocol negotiations are kept local to said modem by said network stack.

51. The method of Claim 45, wherein said network stack disregards any packet received unless there is already a socket connection set up for it.

52. The method of Claim 45, further comprising the step of:
providing a packet sniffer for decoding HTML rating tags, wherein said packet sniffer only passes those pages that are within a preset rating level.

53. The method of Claim 52, wherein said packet sniffer is configured either not to pass unrated pages that do not include a rating tag at all, or to allow said unrated pages to be passed.

54. The method of Claim 52, wherein a rating level can be programmed only via hard wired settings.

55. The method of Claim 45, wherein software applications communicate with said modem core and said network stack via a socket API.

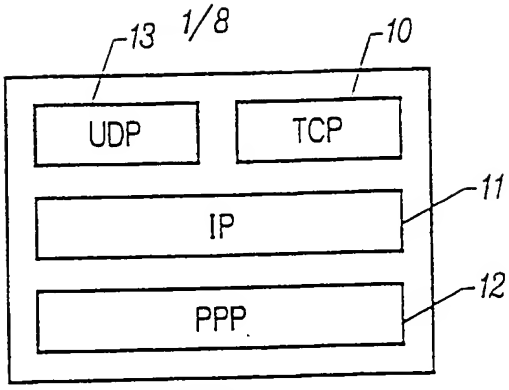
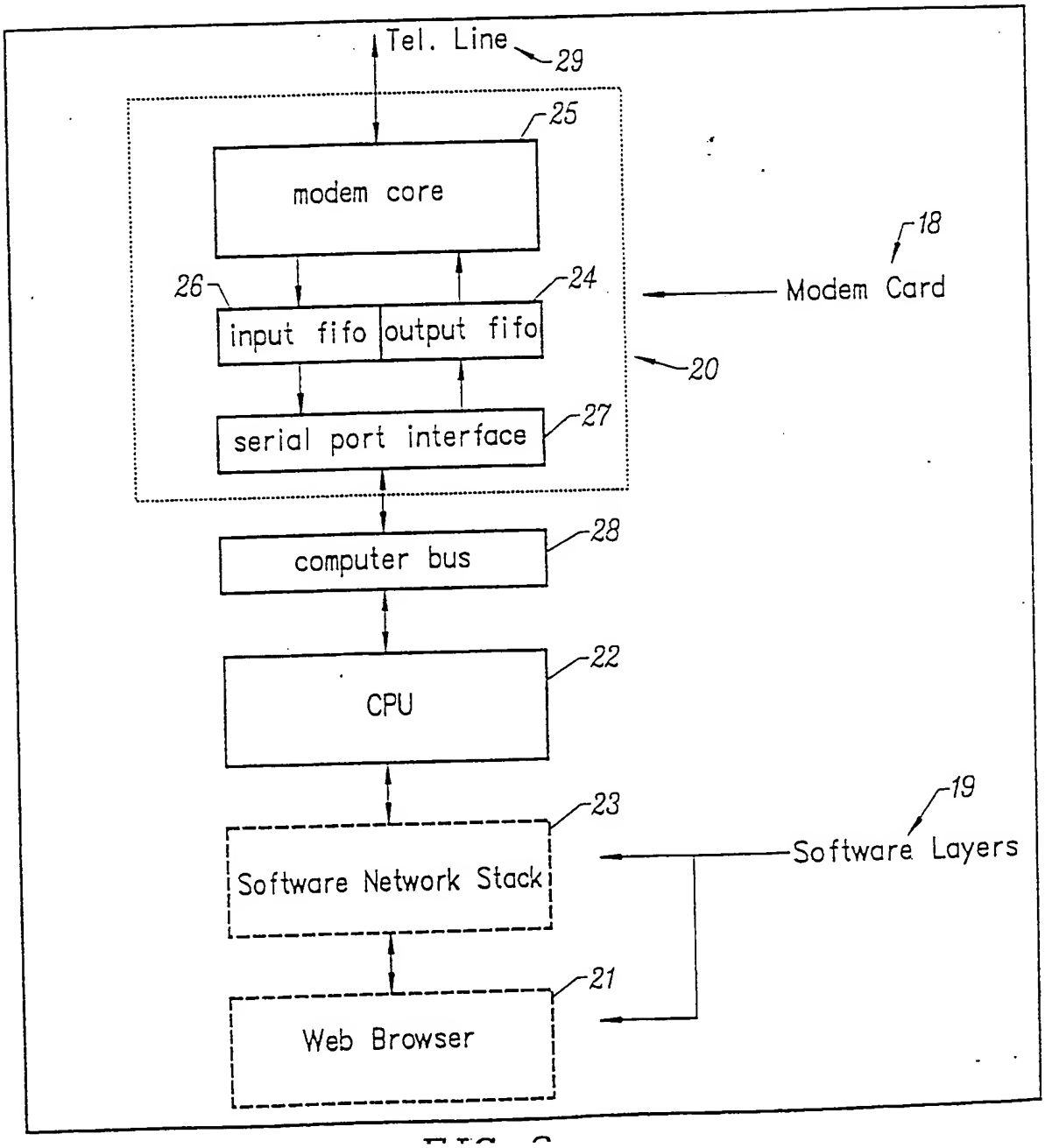


FIG. 1
(PRIOR ART)



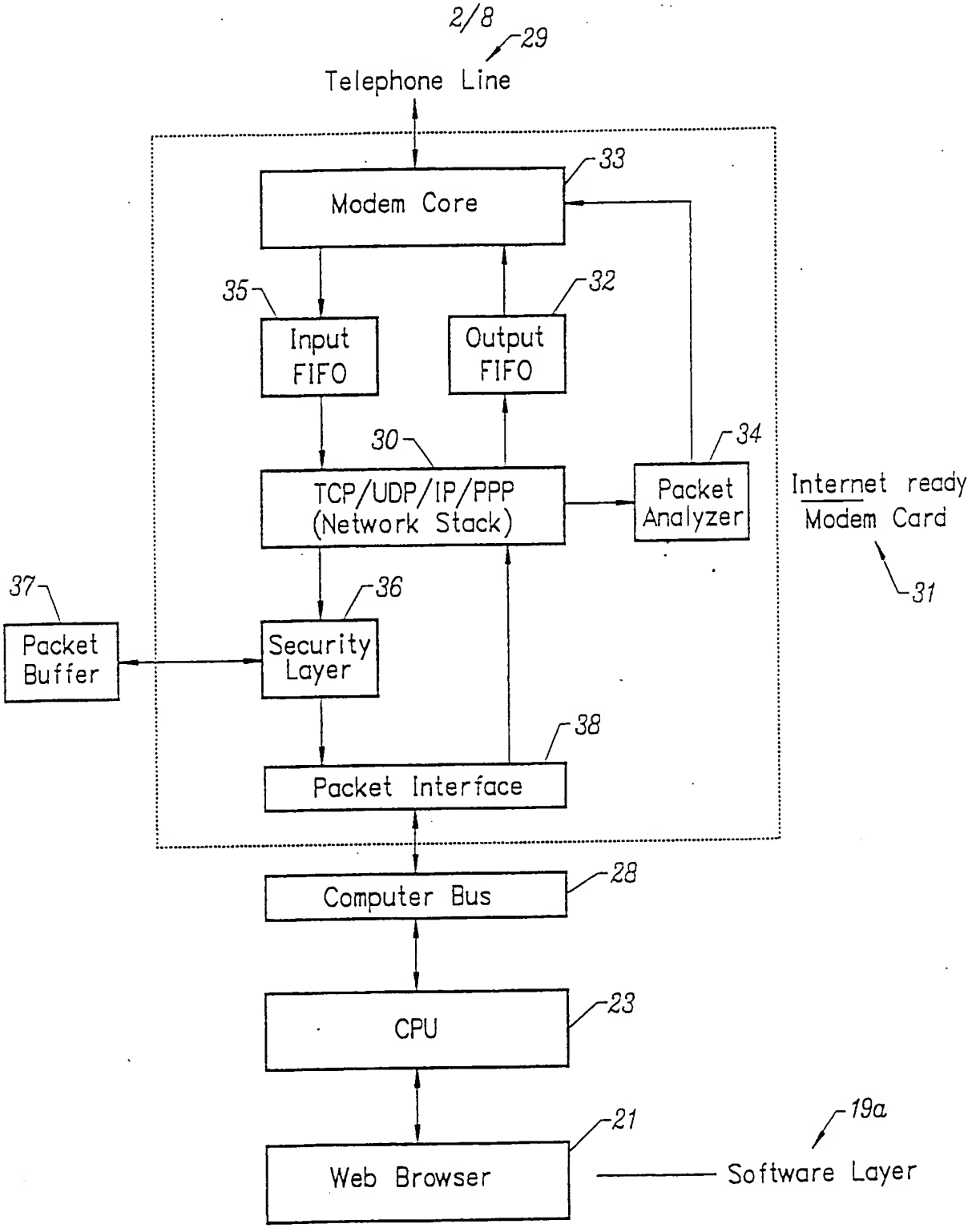

2/8
29
Telephone Line
33
Modem Core
35
32
Input FIFO
Output FIFO
30
34
TCP/UDP/IP/PPP (Network Stack)
Packet Analyzer
Internet ready Modem Card
31
37
36
Packet Buffer
Security Layer
38
Packet Interface
28
Computer Bus
23
CPU
21
19a
Web Browser
Software Layer

FIG. 3

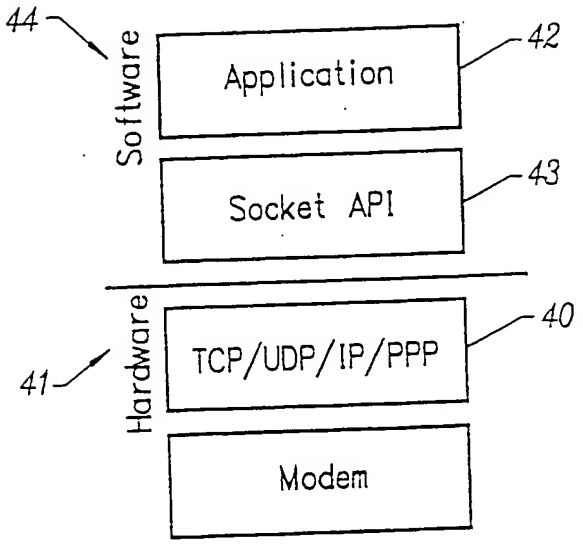

44
Software
Application
42
Socket API
43
Hardware
41
TCP/UDP/IP/PPP
40
Modem

FIG. 4

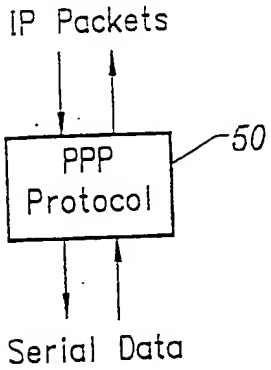

IP Packets
PPP
Protocol
50
Serial Data

FIG. 5

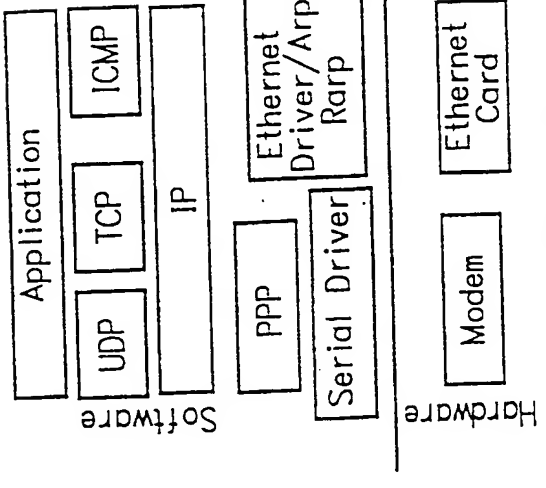


FIG. 6A
(PRIOR ART)

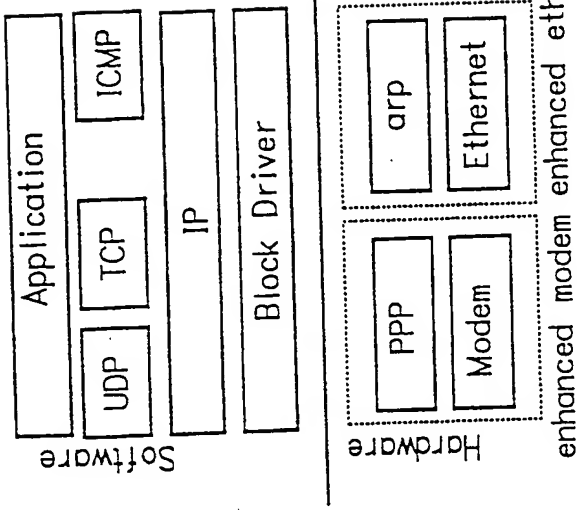


FIG. 6B

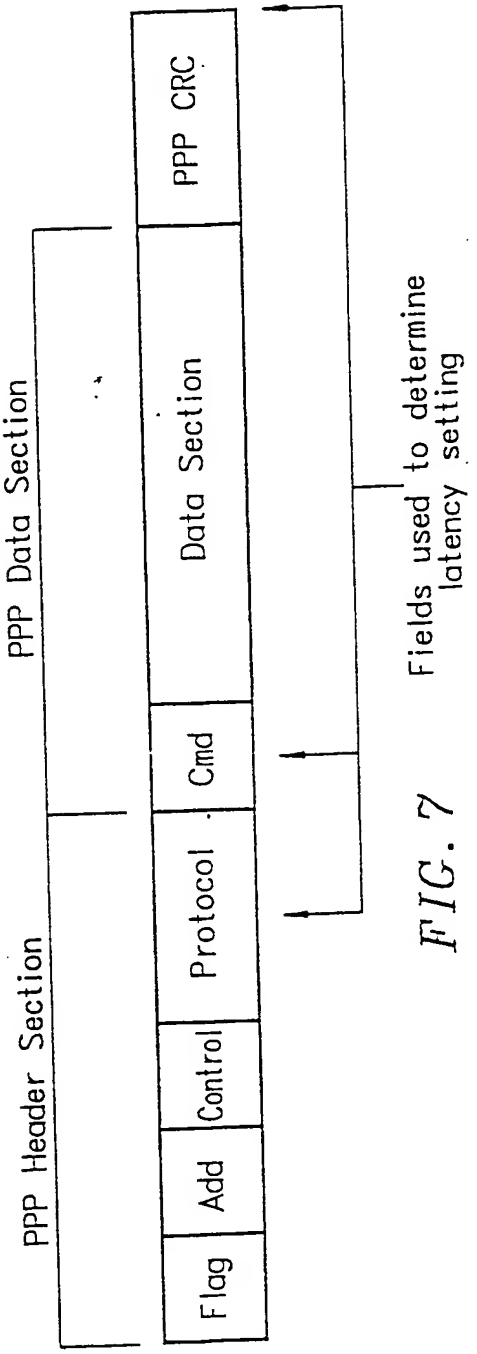


FIG. 7

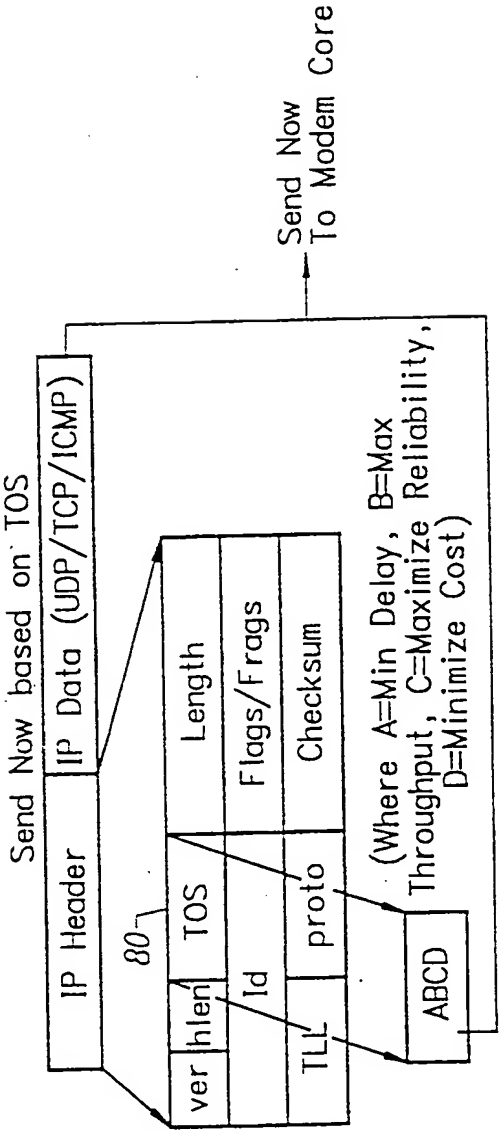

Send Now based on TOS
IP Header
IP Data (UDP/TCP/ICMP)
Send Now
To Modem Core
ver
hlen
TOS
Length
Id
Flags/Frags
TLL
proto
Checksum
ABCD
80
(Where A=Min Delay, B=Max
Throughput, C=Maximize Reliability,
D=Minimize Cost)

FIG. 8

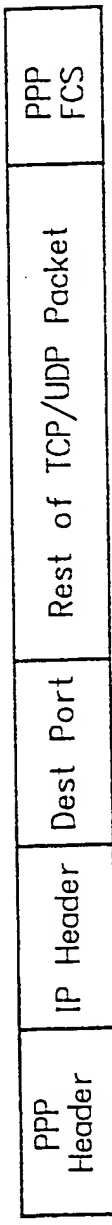

PPP
Header
IP Header
Dest Port
Rest of TCP/UDP Packet
PPP
FCS
Latency
Table
90
Wait Time

FIG. 9

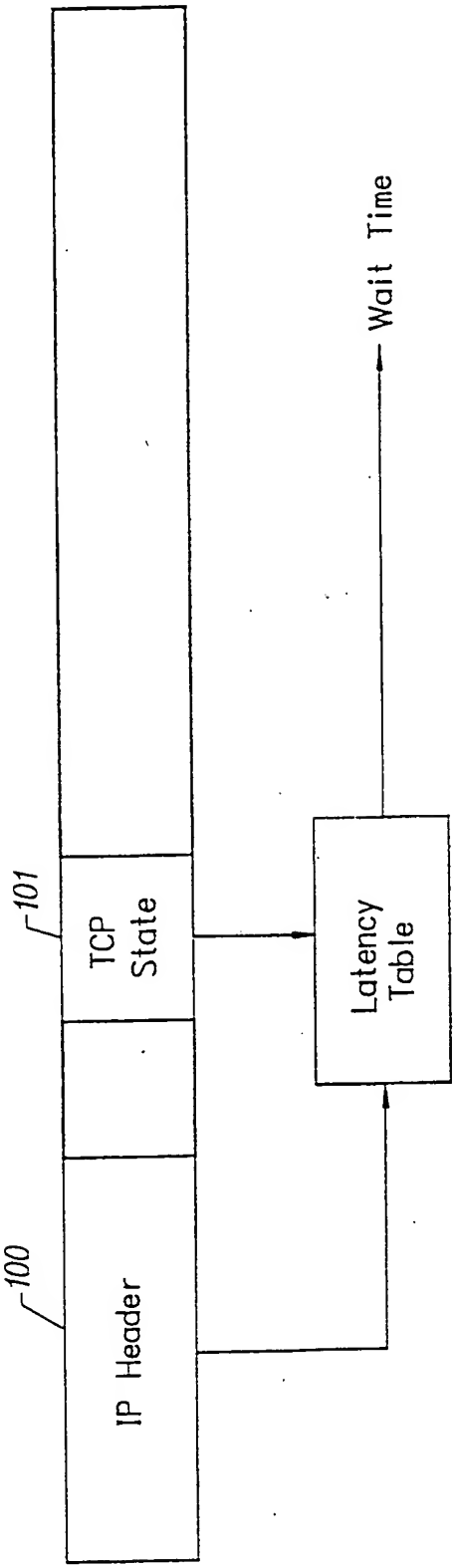


FIG. 10

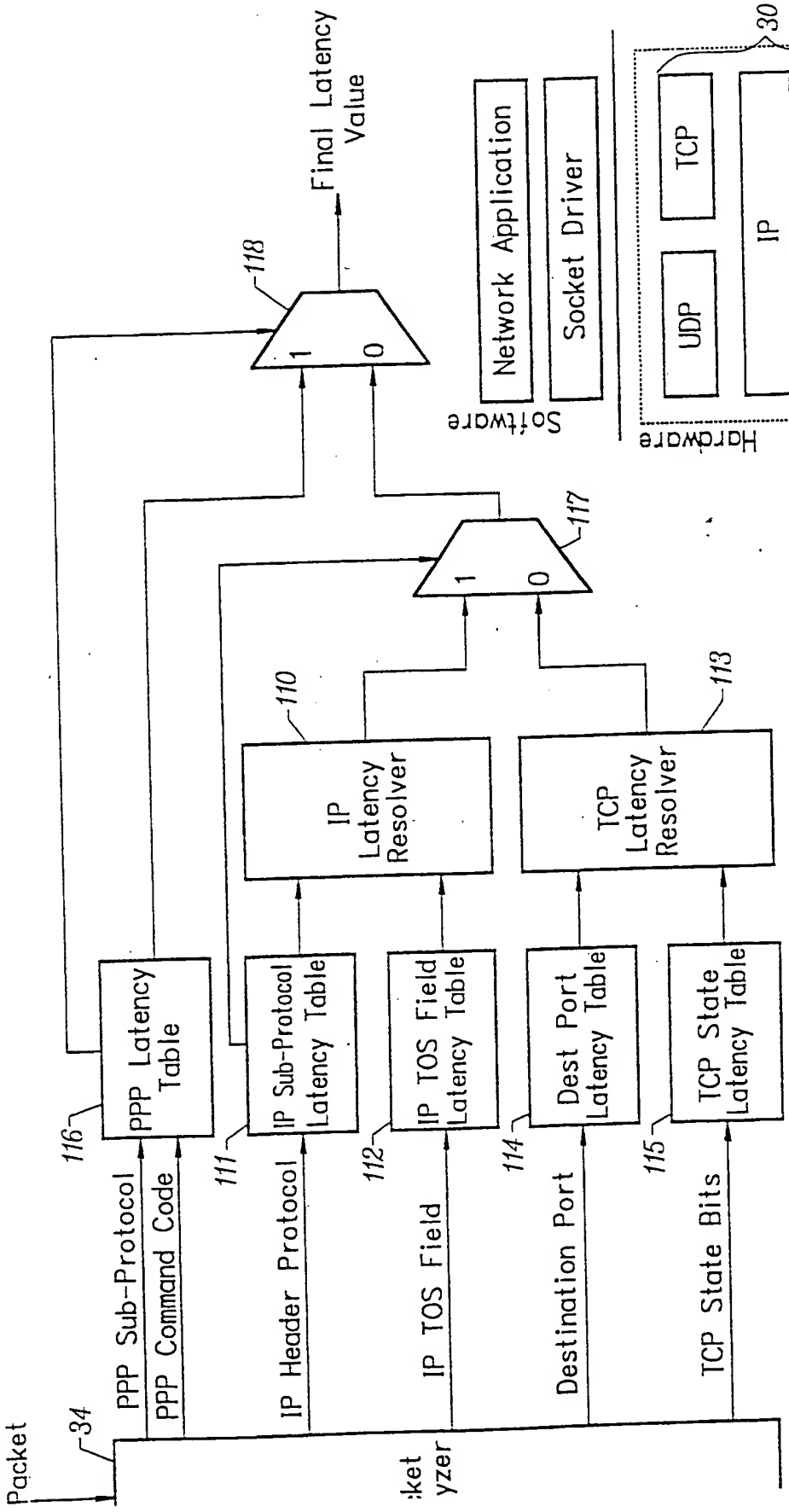

Packet
34
PPP Sub-Protocol
PPP Command Code
IP Header Protocol
IP TOS Field
Destination Port
TCP State Bits
116
PPP Latency Table
111
IP Sub-Protocol Latency Table
112
IP TOS Field Latency Table
114
Dest Port Latency Table
115
TCP State Latency Table
110
IP Latency Resolver
113
TCP Latency Resolver
117
118
1
0
Final Latency Value

FIG. 11

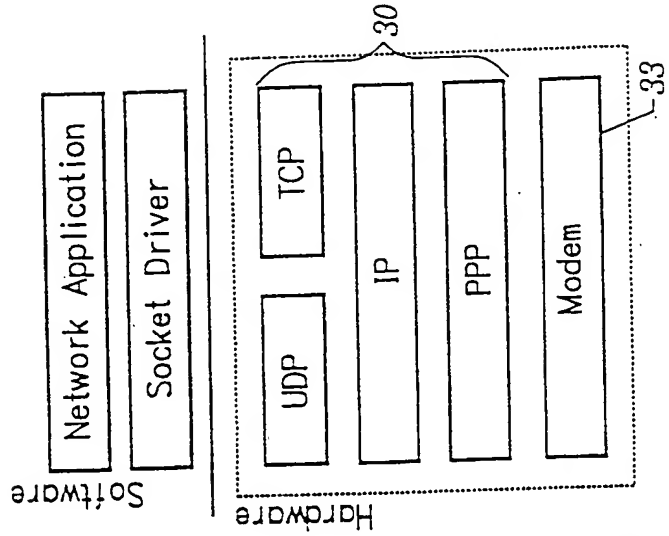

Network Application
Socket Driver
Software
30
UDP
TCP
IP
PPP
Modem
33
Hardware

FIG. 12

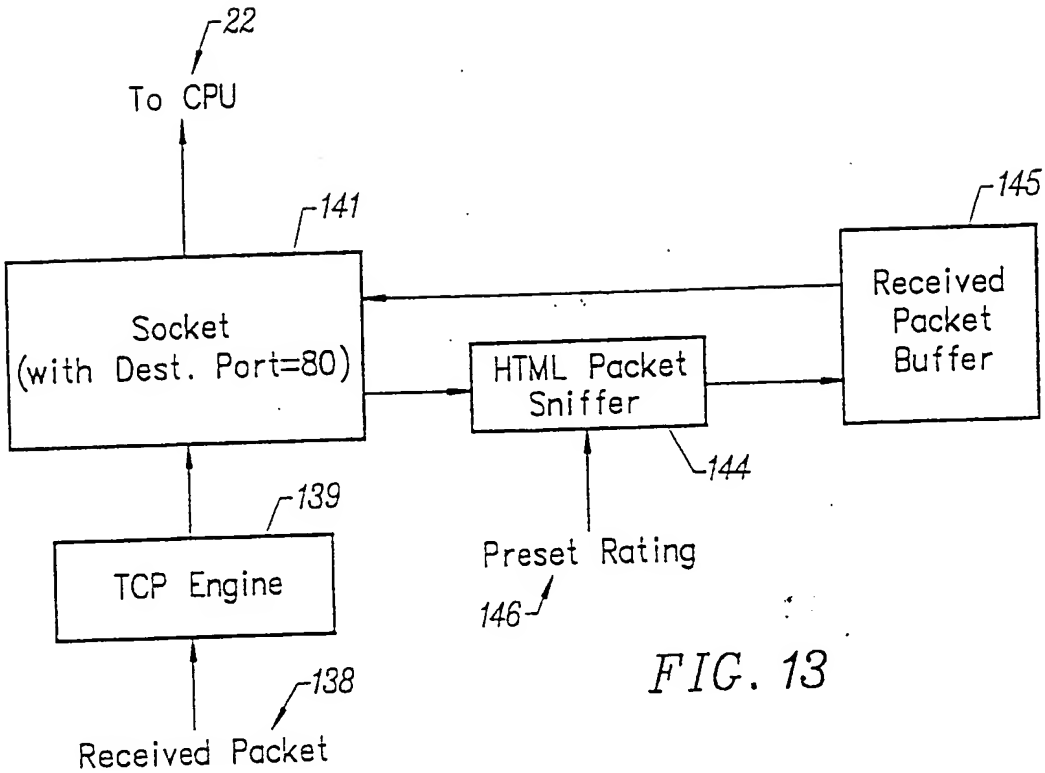

22
To CPU
141
145
Socket
(with Dest. Port=80)
HTML Packet
Sniffer
Received
Packet
Buffer
144
139
TCP Engine
Preset Rating
146
138
Received Packet

FIG. 13

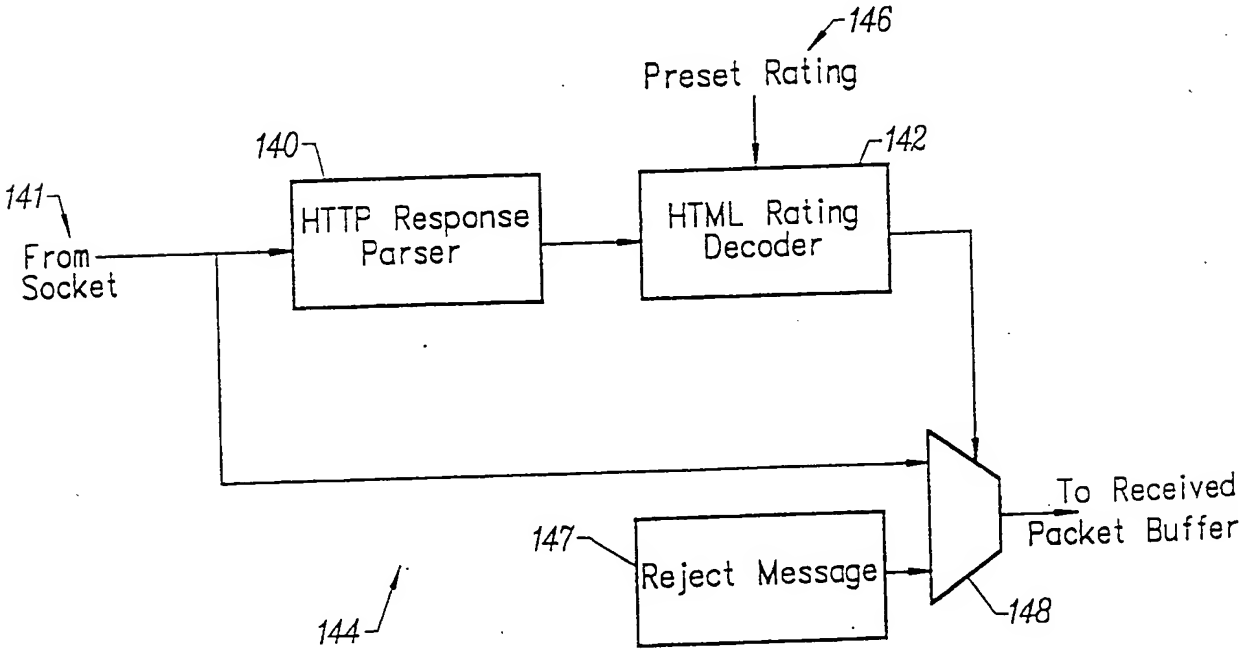

146
Preset Rating
140
142
141
HTTP Response
Parser
HTML Rating
Decoder
From
Socket
To Received
Packet Buffer
147
Reject Message
144
148

FIG. 14

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 95 14356 A (QUALCOMM INC) 26 May 1995 (1995-05-26) figures 1-3 page 5, line 10 -page 7, line 2	1-4, 14-16, 38,45, 50,55
Y	PRESTON D J: "INTERNET PROTOCOLS MIGRATE TO SILICON FOR NETWORKING DEVICES" ELECTRONIC DESIGN, vol. 45, no. 8, 14 April 1997 (1997-04-14), pages 87-90, 92 - 94, XP000730016 ISSN: 0013-4872 page 87, left-hand column, paragraph 1 -page 88, left-hand column, paragraph 2 page 90, right-hand column, paragraph 3 -page 91, right-hand column, paragraph 3	1-4, 14-16, 38,45, 50,55
A	FLOYD S ET AL: "LINK-SHARING AND RESOURCE MANAGEMENT MODELS FOR PACKET NETWORKS" IEEE / ACM TRANSACTIONS ON NETWORKING, vol. 3, no. 4, 1 August 1995 (1995-08-01), pages 365-386, XP000520857 ISSN: 1063-6692 page 367 -page 368	6-8, 17-37, 47-49
A	HIGHLAND H J: "Historical Bits & Bytes" COMPUTERS & SECURITY INTERNATIONAL JOURNAL DEVOTED TO THE STUDY OF TECHNICAL AND FINANCIAL ASPECTS OF COMPUTER SECURITY, vol. 16, no. 5, 1 January 1997 (1997-01-01), page 387-411 XP004096186 ISSN: 0167-4048 page 392, left-hand column, paragraph 2 -page 396, left-hand column, paragraph 6	10,39,51
A	MARCHIORI M: "The limits of Web metadata, and beyond" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 30, no. 1-7, 1 April 1998 (1998-04-01), page 1-9 XP004121445 ISSN: 0169-7552 page 7, left-hand column, paragraph 4 -page 8, left-hand column, paragraph 3 page 1, left-hand column, paragraph 1 -page 2, right-hand column, paragraph 3	11-13, 40-43, 52-54

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9514356 A	26-05-1995	US 5479475 A	26-12-1995
		AU 683706 B	20-11-1997
		AU 1055895 A	06-06-1995
		BR 9405750 A	05-12-1995
		CN 1116481 A	07-02-1996
		EP 0679323 A	02-11-1995
		FI 953418 A	12-07-1995
		JP 8505995 T	25-06-1996
		US 5761204 A	02-06-1998

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04M11/06 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04M H04L G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PORTABLE COMPUTER AND COMMUNICATIONS ASSOCIATION - MODEM STANDARDS COMMITTEE: "The IP Modem Interface Standard" PCCA DRAFT, 'Online! 5 June 1998 (1998-06-05), XP002120914 Retrieved from the Internet: <URL:ftp://ftp.wrq.com/pcca/ipmd0506.pdf> 'retrieved on 1999-10-08! figure 4 page 9, line 3 -page 15, line 21 page 17, line 20 - line 28	1-4,9, 14-16, 38,45, 50,55
A	--- -/--	5,44,46

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
29 October 1999	09/11/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl.	Enaco Holguera